Understanding Security Issues in the DAO Governance Process

Junjie Ma[†‡], Muhui Jiang[‡], Jinan Jiang[‡], Xiapu Luo[‡*], Yufeng Hu[¶], Yajin Zhou[¶], Qi Wang[§*], Fengwei Zhang[§*]
[†]Research Institute of Trustworthy Autonomous Systems, Southern University of Science and Technology, China
[§]Computer Science and Engineering, Southern University of Science and Technology, China
[‡]Department of Computing, The Hong Kong Polytechnic University, China
[¶]Department of Computer Science and Technology, Zhejiang University, China

***Abstract*—The Decentralized Autonomous Organization (DAO) has emerged as a popular governance solution for decentralized applications (dApps), enabling them to manage their members across world. This structure ensures that no single entity can arbitrarily control the dApp without approval from the majority of members. However, despite its advantages, DAOs face several challenges within their governance processes that can compromise their integrity and potentially lead to the loss of dApp assets.**

In this paper, we first provided an overview of the DAO governance process within the blockchain. Next, we identified issues within 3 key components of the governance process: the Governance Contract, Documentation, and Proposal. Regarding the Governance Contract, malicious developers could embed backdoors or malicious code to manipulate the governance process. In terms of Documentation, inadequate or unclear documentation from developers may prevent members from effectively participating, increasing the risk of undetected governance attacks or enabling a small group of members to dominate the process. Lastly, with Proposals, members could submit malicious proposals with embedded malicious code in an attempt to gain control of the DAO. To address these issues, we developed automated methods to detect such vulnerabilities. To investigate the prevalence of these issues within the current DAO ecosystem, we constructed a state-of-the-art dataset that includes 3,348 DAOs, 144 documentation, and 65,436 proposals across 9 different blockchains. Our analysis reveals that many DAO developers and members have not given sufficient attention to these issues. For the Governance Contract, 176 DAOs allow external entities to control their governance contracts, while one DAO permits developers to arbitrarily change the contract's logic. In terms of Documentation, only 71 DAOs provide adequate guidance for their members on governance processes. As for Proposals, over 90% of the examined proposals (32,500) fail to provide consistent descriptions and code for their members, highlighting a significant gap in transparency within the DAO governance process. For a better DAO governance ecosystem, DAO developers and members can utilize the methods to identify and address issues within governance process.

***Index Terms*—Decentralized Governance, Program Analysis, Smart Contracts, Language Models.**

I. Introduction

DECENTRALIZED Autonomous Organization (DAO) is a governance method constructed based on blockchain smart contracts [1]. The DAO ensures that all privileged actions require majority consensus from its members, effectively preventing any single member from taking arbitrary actions.

* Xiapu Luo, Qi Wang, and Fengwei Zhang are the corresponding authors.

Recently, a growing number of decentralized applications (dApps) have adopted DAO as their governance method. For example, Uniswap [2], one of the most valuable Decentralized Exchange (DEX), with a daily trading volume exceeding 500 million dollars [3], employs DAO for its asset management. Additionally, DAO platforms such as XDAO [4], Aragon [5], and DAOhaus [6], which help developers to deploy DAO in minutes, have attracted the interest of thousands of organizations [7]. In particular, XDAO has facilitated the setup of over 16,000 DAOs across various blockchains [4]. According to the analysis [8], the total treasury governed by DAOs exceeds 18.8 billion dollars, with over 2.5 million users. This trend highlights that DAO has become a widely adopted governance method among blockchain developers.

However, the rapid rise in DAOs has brought with it several challenges. Many DAO developers and members fail to pay adequate attention to the issues in the governance process, leading to an increase in attacks targeting DAOs [9], [10], [11], [12], [13], [14], [15], [16], [17], [18]. For instance, a DAO can be attacked through malicious code hidden within a proposal. A notable example is the Beanstalk attack, which resulted in a loss of 182 million dollars [18]. The attacker deceived members into trusting the malicious code in the proposal was benign. Moreover, DAO governance process can be manipulated by developers through hidden backdoor functions controlled by an external entity rather than the governance contract itself. This allows developers to bypass the governance process and take control of DAO assets. An example of this is the VPANDA DAO Rug Pull [19], where a developer illegally transferred over 1 million locked tokens from the contract, gaining over 265 thousand dollars.

Previous studies within the field of DAOs have primarily focused on analyzing DAO activities and issues related to voting in the governance process [20], [21], [22], [23], [24], [25], [26], [27], [28], such as centralized voting power. To the best of our knowledge, no previous work has focused on the issues affecting the entire DAO governance process. Our work fills this gap by conducting a comprehensive study towards the issue within the DAO governance process component **Governance Contract**, **Documentation** and **Proposal** as identified in the section III. The **Governance Contract** governs the entire process, so its integrity must be safeguarded by developers. If not, developers could manipulate the outcome by controlling proposals or altering the contract's

logic. In terms of **Documentation**, DAOs should provide clear and comprehensive instructions to guide members on how to engage in the governance process. A lack of proper documentation may hinder member participation and create opportunities for attackers to push through malicious proposals. For **Proposal**, especially those that involve transferring DAO assets or modifying ownership, the code logic must be clearly defined and fully explained to DAO members. Failure to do so could allow attackers to hide malicious code, resulting in unauthorized control of DAO assets without the members' awareness. To investigate the issues within these components, we address the following 3 research questions, each corresponding to a aspect of the governance process.

RQ1: Does DAO achieve impartial decentralized governance?
RQ2: Does DAO offer sufficient governance process documentation for their members?
RQ3: Does Proposal ensure consistency between descriptions and code?

For RQ1, we verify whether the DAO achieves impartial decentralized governance, ensuring that developers cannot compromise the governance process. First, we perform static analysis of the governance contract to confirm that it correctly implements decentralized governance. Next, we extract the controller addresses of privileged functions to determine whether the contract is self-governed or controlled by developers. Finally, we trace the creation process of the governance contract to ensure developers cannot arbitrarily modify the contract's logic. For RQ2, we investigate whether the DAO provides sufficient guidance to its members for participating in the governance process, thereby encouraging member engagement. We leverage Large Language Models (LLM) with Chain of Thought (CoT) [29] to evaluate if the DAO documentation complies with the 6 requirements outlined in the DAO Model Law [30]. In RQ3, we assess whether the proposals submitted by members exhibit consistent and immutable code behavior that aligns with their descriptions, ensuring that attackers cannot hide malicious code within a normal proposal description. First, we trace the proposal code to verify its immutability. Then, we use a combination of Natural Language Processing (NLP) and LLM to ensure that the actions described by the code are accurately reflected in the proposal descriptions.

The issues we address in this study have not been explored in prior research. Moreover, our investigation covers an extensive dataset of over 3,000 DAOs across 5 platforms and 9 blockchains. Our results show that many DAOs exhibit issues within their governance processes. For the **Governance Contract**, we found that 176 DAOs allow external entities to control their governance contracts, and we identified 1 DAO where the developer can arbitrarily modify the contract's code logic. Regarding the provision of **Documentation** to assist members in participating in governance, only 144 out of 3,348 DAOs provided documentation for their members. Among these, only 71 DAOs offered guidance specifically for their governance process. Given that such documentation is crucial for equipping members with the necessary information to engage in governance, we found that its absence correlates with a significant decline in participation. The average number of proposals decreased from 34 to 9, while the average voting participation drops from 3,342 to just 33. For **Proposal**, we found that only 3,018 out of 35,518 proposals do mention all their code actions in the proposal, such as which functions will be invoked and how many tokens will be transferred. To assess the effectiveness of our approach in detecting real-world malicious proposals, we tested it against 13 malicious proposals from recent governance attack incidents. Our method successfully detected all of these attacks.

We hope that our paper can guide developers in deploying and maintaining their DAOs in a more secure and comprehensive manner, while also raising awareness among members about potential risks within DAO governance.

Our contributions can be summarized as follows.

Public Dataset. We collected 3,348 actively used DAO implementations, 144 documentation, and 35,518 proposals across 9 popular blockchains. Our dataset included famous DAOs such as Uniswap [2] and Compound [31], as well as DAOs from platforms like Aragon [5]. The collected data will be released for further research.

Comprehensive Study. We conducted an in-depth study of DAO implementations, addressing 3 key research questions related to each component of DAO governance processes.

Insightful Findings. Our study revealed significant security issues in current DAO implementations. We found that around 5% of DAOs are controlled by unknown entities, over 94% lack any documentation, and more than 92% of proposals fail to explain the actions their code will execute.

II. Background

A. Decentralized Autonomous Organization

Decentralized Autonomous Organization (DAO) is first introduced by Ethereum white paper [1]. DAO utilizes smart contracts to enable collective control of the organization by all its members. Currently, there are two types of DAO governance [32], [33]: on-chain governance and off-chain governance. On-chain governance requires all the governance processes to be conducted on the blockchain by smart contracts, including proposing proposals, voting, and executing. On the contrary, in off-chain governance, the decision-making process (e.g., proposing proposals or voting) is performed outside the blockchain. The execution process is carried out manually by the DAO developer, granting it complete control over the DAO contracts. We exclude off-chain governance from our scope as it contravenes the requirements in the DAO definition [1] and the DAO Model Law [30], which mandate governance process to be executed on the blockchain.

B. DAO Platform

The DAO platform is designed to provide DAO developers with the tools to easily create their own DAOs. Developing a DAO requires advanced programming and blockchain knowledge. Current DAO platforms such as XDAO [4], Aragon [5], DAOhaus [6], and DAOstack [6] offer comprehensive assistance in DAO creation. This support ranges from deploying contracts to building websites. With these platforms, developers can create their own DAO in minutes.

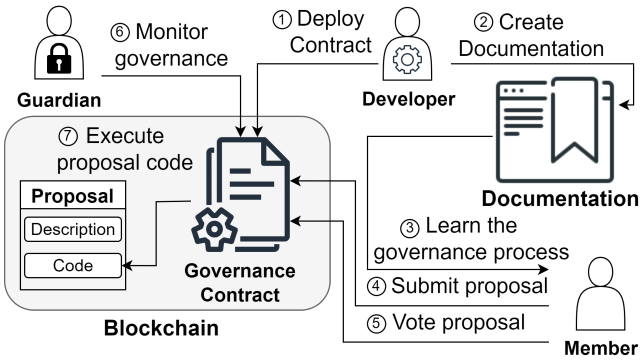


Fig. 1: The DAO Governance Process.

C. DAO Model Law

The DAO Model Law [30], a type of Model Law [34], aims to bridge the gap between DAOs and traditional regulatory frameworks, which have yet to adapt to the new organizational structures enabled by blockchain. The DAO Model Law stipulates rules applicable to both on-chain smart contracts and off-chain documentation. Once these rules are adhered to, DAOs and their members can achieve legal certainty.

III. DAO Governance Process

We provide a comprehensive overview of the DAO governance process, as shown in Figure 1.

Participants. The participants in the DAO governance process fall into one of three roles: *developer*, *member*, and *guardian*. The first role, *developer*, is involved in the development of the DAO's smart contracts and interface. He is responsible for ① deploying the governance contracts to the blockchain network, as well as ② creating the documentation for the DAO. The second role, *member*, is a blockchain user who learns the governance process by ③ reading the documentation. He can participate in DAO governance by ④ submitting or ⑤ voting for a proposal via the DAO governance contract [31]. The last role is the *guardian*, a specific blockchain user tasked with ⑥ monitoring the DAO governance process. If the *guardian* detects malicious proposals targeting DAO governance, he has the authority within the governance contract to cancel such proposals.

Governance contract. The governance contract controls the governance process, storing all the proposals and votes from members. It provides functions that allow members to submit new proposals, vote on these proposals, and execute the code within the proposals. The governance contract should be configured as the only way to change the DAO contracts.

Proposal. Proposal refers to a formal submission to the governance contract made by a member to suggest changes to the DAO (i.e., funding request, contract parameters configuration). Typically, as shown in Figure 14, the proposal encompasses two elements: *description* and *code*. The *description*, penned in natural language, outlines the intent of the proposal. It provides members with information regarding the proposal code and its reason. The *code* contains the code the governance contract will execute if the proposal passes. It refers to the technical implementation of the proposal.

```
modifier onlyGovernance() {
    require(msg.sender == address(this), "Governor:onlyGovernance");
    _;
}

/// @notice Sets the voting delay
/// @param newVotingDelay New voting delay
function setVotingDelay(uint256 newVotingDelay) public onlyGovernance {
    uint256 oldVotingDelay = _votingDelay;
    _votingDelay = newVotingDelay;
    emit VotingDelayUpdated(oldVotingDelay, newVotingDelay);
}
```

Fig. 2: A simplified privilege function restriction requires the function caller to be the governance contract.

Governance process. Managing and implementing changes within a DAO relies on the *governance process*. This is achieved by submitting proposals to the governance contract and conducting votes on these proposals. If a proposal passes the voting process, the code within it is executed by the governance contract to implement the changes towards the DAO. This ensures that the majority of the DAO members approve all the changes. The governance process begins at a *member* ④ submitting a proposal to the governance contract. Then, a *member* can ⑤ cast a vote for the newly submitted proposal. A proposal is passed when it has received sufficient voting power in support from members. If the *guardian* does not identify the proposal as a malicious one ⑥, the code within the proposal will be executed by the governance contract ⑦.

Documentation. Considering the complexity of the governance process, the *documentation* should provide complete guidance on the governance process. This encompasses delivering detailed information on becoming a DAO member, providing step-by-step guides to participate in the governance process, and outlining the existence of *guardian*.

IV. Approach

A. Research Questions

We examine the issues within each component of the governance process - **Governance Contract**, **Documentation**, and **Proposal** - with the following research questions.

The **Governance Contract** controls the entire governance process. According to the definition [1], it is essential for achieving impartial and immutable in decentralized governance, preventing developers from arbitrarily manipulating the results. As shown in Figure 2, the $setVotingDelay$ function, which is designed to adjust the voting duration, includes a modifier named $onlyGovernance$. This modifier ensures that the function can only be invoked when the caller's address ($msg.sender$) matches the address of the governance contract itself. This restriction indicates that the function operates under the authority of the governance contract. However, if such functions are controlled by developers, they could potentially manipulate parameters such as the voting duration and the required voting power for proposals, thereby influencing the governance process results.

Thus, we propose RQ1 to examine whether the Governance Contract achieve impartial decentralized governance.

RQ1: Does DAO achieve impartial decentralized governance?

As for the **Documentation**, each DAO should provide detailed instructions for its members on how to participate in the governance process, with an emphasis on disclosing its critical aspects. The absence of proper governance documentation can discourage members from participating, as they would need to rely on reading the governance source code to understand the process. This scenario can lead to governance outcomes being controlled by only a small group of members. For instance, in the Synthetify DAO governance attack on October 17, 2023 [35], an attacker submitted a malicious proposal aimed at seizing control of the DAO's assets. Due to the lack of governance documentation, none of the members actively participated in the process, and as a result, no one vetoed the malicious proposal during the 7-day voting period, leading to a loss of 230 thousand dollars.

In RQ2, we assess whether DAOs provide sufficient documentation to guide members in participating in the governance.

RQ2: Does DAO offer sufficient governance process documentation for their members?

As for the **Proposal**, attackers can submit malicious proposals to gain control of the DAO or misappropriate its assets by embedding malicious code within the proposal. They may deceive members by providing misleading descriptions that make the code appear legitimate.

In RQ3, we investigate the consistency between proposal descriptions and the underlying code to prevent malicious members from submitting deceptive proposals that disguise harmful actions as legitimate ones.

RQ3: Does Proposal ensure consistency between descriptions and code?

B. Data Collection

In this section, we aim to collect and construct a comprehensive DAO dataset for our analysis, which includes the DAO name, governance contract, website, documentation, and proposals related to the DAO governance process. However, gathering this data presents several challenges.

First, there is no existing comprehensive dataset that encompasses all relevant DAO information. Second, current DAO data platforms fail to provide documentation and include only a limited number of DAO websites. Third, not all DAO platforms offer APIs for retrieving proposals.

To address these challenges, we outline our data collection methods for each type of data as follows:

DAO Name and Governance Contract. To gather as comprehensive a list of DAOs as possible, we collect DAO names and corresponding governance contract addresses from platforms mentioned in previous studies [36], such as Aragon [5], DAOhaus [6], and DAOstack [37]. Additionally, we include DAOs from two currently popular platforms, XDAO [4] and Tally [38]. To account for self-developed DAOs that do not belong to these platforms, we also collect DAO information from the DAO analytics website DeepDAO [8].

Using the APIs provided by these platforms, we collect DAO names and governance contract addresses. Since the data from DeepDAO may include DAOs from other platforms, we remove duplicates, treating two DAOs with identical contract addresses as the same entity.

TABLE I: Types of DAOs, along with their corresponding quantity, website, documentation, and proposal in the database.

DAO Type	DAO (original)	DAO (filtered)	Website	Documentation	Proposal
XDAO	16,018	2,357	105	52	29,586
Aragon	2,939	630	51	24	21,023
Tally	1,256	266	69	55	8,999
DAOhaus	278	62	8	5	1,827
DAOstack	41	30	8	5	2,419
Self-developed	3	3	3	3	1,582
Total	20,535	3,348	244	144	65,436

Our dataset focuses on DAOs operating on EVM-compatible chains with a Total Value Locked (TVL) exceeding 50 million dollars. These chains include Ethereum [39], BSC [40], Polygon [41], Fantom [42], Gnosis [43], Avalanche [44], Arbitrum [45], Cronos [46], and Optimism [47]. We collected data from these sources until September 1, 2024. The results are shown in Table I. In total, we gathered data on 30,535 DAOs. To remove unused or experimental DAOs that might introduce bias into the results, we filtered the dataset by selecting those DAOs with at least four proposals and eight voting records from at least two different members. This process resulted in a final dataset of 3,348 actively used DAOs.

To ensure the completeness of our dataset, we cross-checked it with the top 20 DAOs listed on CoinMarketCap [3]. The results confirm that all top 20 DAOs, including Uniswap [2] and Compound [31] from Tally, as well as Curve [48] and MakerDAO [49] from DeepDAO, are included in our dataset.

Documentation. To collect the documentation, we first gathered DAO websites using platform APIs and data from DeepDAO. For DAOs without a listed website, we queried their public name tag [50] from blockchain scanners to determine if the governance contract was linked to a DAO website. We then used Selenium [51] to crawl through the DAO websites to retrieve documentation. Specifically, we focused on links containing keywords such as "whitepaper" or "doc." If no such specific links were found, we archived the entire website for further analysis. As shown in Table I, we found that only a small proportion of DAOs, specifically 244 out of 3,348, provide a website. However, we discovered that 100 of these websites were either offline or had expired domain names, leaving only 144 operational DAO websites.

We hypothesize that this may be due to the lack of website maintenance, with only popular DAOs able to create and sustain their websites. To validate this, we examined DAOs with a Total Value Locked (TVL) exceeding 20 million dollars, based on data from CoinMarketCap [3]. We found that all 11 DAOs in this category still maintain their websites.

Next, we analyzed whether the low rate of online websites could be due to DAOs being out of service. We defined a DAO as out-of-service if it had not submitted any new proposals within a year. Our findings revealed that 2,477 out of 3,348 DAOs are still active, while 871 are no longer active. Interestingly, 54 out of the 871 inactive DAOs still maintain their websites, whereas only 90 out of the 2,477 active DAOs have maintained their websites.

Proposal. To retrieve the proposals, for platforms such as Aragon, DAOhaus, and DAOstack that provide APIs, we utilized these APIs to download all the proposals associated with each DAO. For other DAOs that do not provide an API, we extracted proposal creation event logs [52] from

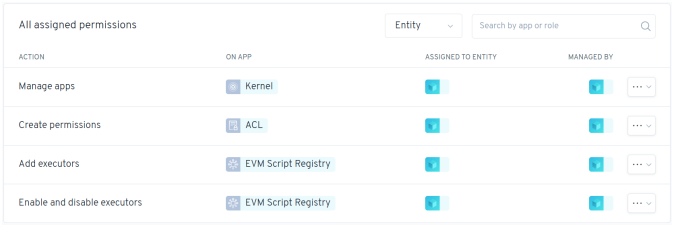


Fig. 3: The DAO from Aragon, despite its claims of being governed by DAO, does not provide functionality for its members to propose or vote on proposals.

the governance contract addresses and retrieved the proposal information directly from these logs.

V. Does DAO achieve impartial decentralized governance?(RQ1)

In this section, we examine whether existing governance contracts implement impartiality in decentralized governance. Specifically, we assess 3 key aspects of the governance contract: correctness, self-governance, and immutability. First, for correctness, we evaluate whether the governance contract is capable of facilitating a decentralized governance process. A failure in this capability would violate the core principles of a DAO. Second, we assess self-governance to ensure that developers cannot compromise governance outcomes by invoking privileged functions, which would undermine decentralized control. Finally, for immutability, we investigate whether the governance contract's code can be altered by developers, as this could allow manipulation of the governance process.

A. Correctness of Governance Contract

As stipulated by the DAO definition [1], [30], a DAO's governance must be decentralized. This requires the governance contract to effectively facilitate decentralized governance. If the governance contract lacks the ability to ensure this, it would violate the core principles of a DAO. As illustrated in Figure 3, the DAO *0x022f...528a* from Aragon claims to be a DAO. However, it does not include the necessary voting functionality, preventing members from proposing or voting on proposals. Consequently, all DAO assets and privileges are controlled solely by the DAO developers, undermining the principles of decentralization.

Approach. To evaluate whether a DAO has correctly implemented decentralized governance within its Governance Contract, we employ different methods depending on the type of DAO. For DAOs from platforms XDAO, Aragon, DAOhaus, and DAOstack, it is mandatory for them to use the template governance contracts provided by their platforms [21]. We first conduct a manual analysis to verify whether the template governance contracts from these platforms correctly implement decentralized governance. Next, we confirm whether each DAO has adopted the provided template governance contract. To verify whether a DAO's governance contract matches the template, we trace the creator of the governance contract and compare it to the deployer address listed in the platform's deployment guide. For governance contracts with different creator addresses, we check whether the bytecode of the DAO's governance contract matches the template contract. If either of these checks passes, we determine that the DAO has correctly implemented decentralized governance.

For DAOs from Tally, the developers are allowed to add new functions based on the template contract provided by OpenZeppelin [53] or Compound [31]. We can not directly compare the bytecode of these contracts to ascertain if it is the same as the template contract. Thus, we check whether the governance contract includes the three governance functions from the template contract (i.e., *Propose*, *Vote*, and *Execute*) as required by the DAO Model Law [30] as well as the template contract from OpenZeppelin and Compound. (1)*Propose.* A member can submit a proposal by invoking this function. (2)*Vote.* For a proposal recorded in the contract, members have the ability to cast their votes using this function. (3)*Execute.* The function can execute the code of the proposal. If a DAO's governance contract includes all 3 required functions, we conclude that it adheres to the template contract. We use EVM CFG BUILDER [54] to extract the bytecode of each function from the governance contract. We then compute the similarity of bytecode between the governance contract's functions and the template functions by calculating hypervectors of n-grams (n=5) of opcodes and comparing them using the Jaccard similarity [55]. If the similarity score exceeds 0.8 [56], we consider the functions to be equivalent. To account for discrepancies caused by different versions of Solidity compiler, we recompile the contracts using each major Solidity version. If the target function matches any version of the template contract function, we conclude that the DAO's governance contract includes the required function.

For other DAOs, if the governance contract is open-source, supported by documentation, and aligns with decentralized governance principles, we infer that the DAO has achieved decentralized governance. Otherwise, we check whether the governance contract is similar to the contract provided by the platform or includes functions similar to those in the template contract, using the same approach outlined above.

Result. The results, as shown in Table II, indicate that all the analyzed DAOs implement decentralized governance. However, during the evaluation of our method (Appendix A), we detected that some DAOs on certain platforms do not enforce decentralized governance in their governance contracts. This is due to platforms giving developers the discretion to either include or exclude decentralized governance during DAO creation. To uphold the principles of decentralization, platforms may want to consider making decentralized governance a mandatory feature for developers.

B. Self-governance of Governance Contract

All privileged functions within the governance contract should be controlled by the governance contract itself to prevent any potential violations from developers. The privileged function is defined as a function that can be executed only by a privileged address [57], [58]. However, if the governance contract does not govern certain DAO functions, this could lead to security vulnerabilities. In the case of the governance contract 0x41E6.....7a42 from the DAO "mini dao," shown

TABLE II: Numbers of DAOs that achieve decentralized governance (DG), along with those where privileged functions are controlled by the governance contract or other entities.

DAO Type	With DG	Without DG	Governance	Other
XDAO	2,357	0	2,286	71
Aragon	630	0	612	18
Tally	266	0	179	87
DAOhaus	62	0	62	0
DAOstack	30	0	30	0
Self-developed	3	0	3	0
Total	3,348	0	3,172	176

```
function _setVotingPeriod(uint256 newVotingPeriod) public nonPayable {
    require(msg.sender == _admin, Error('GovernorBravo::_setVotingPeriod: admin only'));
    _votingPeriod = newVotingPeriod;
    ...
}

function _setProposalThreshold(uint256 newProposalThreshold) public nonPayable {
    require(msg.sender == _admin, Error('GovernorBravo::_setProposalThreshold: admin only'));
    _proposalThreshold = newProposalThreshold;
    ...
}
```

Fig. 4: The decompiled governance contract from **mini dao** shows that the developer controls privileged functions (*setVotingPeriod* and *setProposalThreshold*), enabling him to control proposal voting duration and required voting power.

in Figure 4, these functions are controlled by an *admin*, an Externally Owned Account (**EOA**) choosed by the developer, rather than by the governance contract itself. As a result, the developer could manipulate the process by adjusting the voting delay to ensure only they can vote, or by setting an unreasonably high proposal threshold to cancel any unwanted proposals.

Approach. Thus, we examine whether there are privileged functions within the governance contract that are controlled by external entities instead of the governance contract itself.

For DAOs from platforms XDAO and Aragon, these DAOs use a standardized contract for both governance logic and access control. Additionally, these platforms provide official APIs [4][5] to query the governor of the privileged functions. By using these APIs, we can determine whether the governance contract controls all privileged functions.

For other DAOs, inspired by previous studies [57][58], we apply static analysis of the governance contract bytecode to identify privileged functions and extract the privileged addresses associated with these functions. Specifically, to identify privileged functions, we analyze whether a function checks the caller's address, obtained via the $CALLER$ opcode, against a specific address from contract storage using the EQ opcode. This comparison is used to determine the $jump$ target. We then extract the address and compare it with the governance contract address to ascertain if they match.

Result. As demonstrated in Table II, the majority of DAOs, particularly those on platforms DAOhaus and DAOstack, strictly follow the requirement that all functions within the governance contract should be governed by the governance contract itself. However, 176 governance contracts retain certain privileged functions that are not governed by themselves. As indicated in Section VI, most DAOs fail to explain the existence of guardians. Thus, it is hard for members to classify whether these functions are potentially backdoors or designated for guardians to protect the governance process.

```
function _upgradeToAndCall(address newImplementation, bytes memory data, bool forceCall) internal {
    ...
    _functionDelegateCall(newImplementation, data);
}

function _functionDelegateCall(address target, bytes memory data) private returns (bytes memory) {
    require(AddressUpgradeable.isContract(target), "Address: delegate call to non-contract");
    (bool success, bytes memory returndata) = target.delegatecall(data);
    return AddressUpgradeable.verifyCallResult(success, returndata, "Address: low-level delegate
call failed");
}
```

Fig. 5: The governance contract of DAO based on OpenZeppelin, created using the *CREATE2* chain. The contract allows developers to indirectly destroy it by executing a delegate call to another contract that contains the *SELFDESTRUCT*.

C. Immutability of Governance Contract

After the Constantinople update [59], the EVM introduced a new opcode, $CREATE2$, which allows a smart contract to be deployed at a predetermined address [60]. This can be exploited as an attack vector, as it enables contract developers to modify the contract code after deployment while keeping the contract's address unchanged [13], [61], as demonstrated in Appendix B. Unlike traditional proxy contracts [62], where a developer must first deploy an intermediary contract that stores the governance contract's address if they wish to make the contract upgradeable, they can later deploy a new governance contract and update the proxy contract's address through a transaction. As a result, any changes or upgrades to the governance contract can be tracked through the proxy contract's transaction history. In this case, while the proxy contract's address remains the same, the actual governance contract address changes with each update. However, with the $CREATE2$ method, as discussed in Appendix B, developers can secretly re-deploy the governance contract by first destroying the contract and then redeploying it at the same address. This allows the governance contract's address to remain unchanged, making it difficult for regular blockchain users to detect that the contract code has been altered or upgraded unless they thoroughly trace all related transactions. In contrast to the proxy contract approach, $CREATE2$ maintains the same address despite any changes to its logic. As illustrated in Figure 5, the governance contract of DAO *0xfbac...41b6*, built on OpenZeppelin and deployed via the *CREATE2* chain, contains a function named *_functionDelegateCall*. This function allows developers to delegate calls to external contracts. By exploiting this functionality, developers can indirectly destroy the governance contract to invoke a *SELFDESTRUCT* opcode hidden within an another contract.

Approach. We first define the Contract Creation Chain (CCC) of a governance contract as follows: Given a governance contract address G, we trace its contract deployment transaction. If it is deployed by a contract C_0, we add it to the CCC. We then trace the creator of C_0, designated as C_1, and continue this process until we find a contract that is created by an EOA address E. The $CCC(G) =< G, C_0, C_1, ..., E >$ shows the governance contract G is created from a chain of contracts that extend from C_0 to E.

To determine whether a given governance contract address G is at risk from $CREATE2$,, we first construct its Contract Creation Chain (CCC). For each contract C_i in $CCC(G)$, we check whether the contract can self-destruct

using the $SELFDESTRUCT$ opcode to erase its own code. However, a potential attacker could conceal the opcode within a different contract and indirectly execute it using the $DELEGATECALL$ to destroy the original contract. Hence, if a contract contains the opcode $SELFDESTRUCT$ or $DELEGATECALL$, we infer that it can destruct itself. Subsequently, in order to check whether contract C_i is created by $CREATE2$, we trace the opcodes used during the contract deployment transaction. If the $CREATE2$ opcode is used to create C_i, we deem that C_i is created by $CREATE2$. We adopt Tenderly API [63] to access the executed opcodes from the deployment transaction of contract C_i. Finally, if we determine that contract C_i was created by $CREATE2$ and that all preceding contracts in the chain can self-destruct, we conclude that contract C_i is under the threat of $CREATE2$.

Result. We identified one DAO from Tally, associated with the governance contract $0xfba...b6$, which was created using the $CREATE2$ opcode and utilizes the $DELEGATECALL$ opcode to interface with external contracts. Notably, contracts deployed via $CREATE2$ can be destroyed by developers and redeployed at the same address.

After analyzing the governance contract, we find that the vulnerability might have been introduced accidentally by the developer. The governance contract includes an internal function, *functionDelegateCall*, which allows external contracts to be called with a *DELEGATECALL*. Thus, a passed proposal containing the *SELFDESTRUCT* opcode can lead to the contract's destruction. This would then allow the developer to redeploy the governance contract using $CREATE2$ and $CREATE$. While this issue could be unintentional, we cannot ignore the possibility that a malicious DAO developer could exploit it to execute an attack.

Answer to RQ1: Among the 3,348 DAOs analyzed, we found that 176 could potentially be manipulated by developers, and one DAO's governance contract code logic can be directly altered by its developer. This indicates that not all currently active DAOs can be trusted to achieve impartial decentralized governance.

VI. Does DAO offer sufficient governance process documentation for their members?(RQ2)

The documentation is expected to provide a comprehensive overview of the DAO, detailing the governance process and how members can interact with it. Given that the DAO Model Law [34], as referenced in Section II-C, is the only harmonized regulatory framework prescribing specific rules for DAO documentation and its participants [64], we thoroughly reviewed the DAO Model Law and extracted all relevant requirements concerning DAO documentation, which we summarized into six key rules. 1) *Member Participation.* The documentation should provide guidelines on how blockchain users can become DAO members and participate in governance, as well as the participation rights in the governance process. 2) *Member Exit.* Apart from participating in DAO, the documentation should also describe the steps a member needs to follow to exit the DAO, whether in a voluntary or involuntary way. 3)

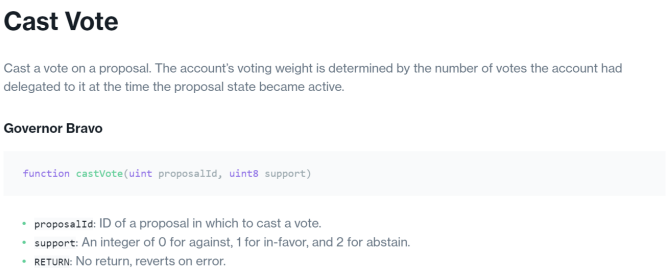


Fig. 6: The Compound governance documentation provides DAO members with guidance on how to vote for proposals.

Voting Power. The documentation should clearly explain how voting power is calculated and distributed among members, as voting power determines the weight of a member's vote. Failing to explain voting power could discourage member participation in voting or, conversely, enable a member to accumulate excessive voting power, potentially allowing him to arbitrarily control the result of voting. 4) *Minority Protection.* The documentation should explicitly state if it includes any provisions for protecting the minority rights of its members. This is crucial because minority members may need to raise disputes against specific decisions, particularly in situations where a single member controls the majority of voting power. 5) *Governance Process Guide.* A detailed guide to the governance process is necessary for members. For instance, the step-by-step instructions for submitting proposals and casting votes. 6) *Appointment of Guardian.* The appointment of a guardian is crucial to alleviating security concerns among members. Given the significant privileges the guardian holds, such as controlling the privilege functions in the governance contract, their role should be disclosed in the documentation.

Considering most of the members are not able to reliably and accurately extract information from the on-chain DAO contract code, it's vital that the DAO presents this information in the transparent, publicly accessible document. For example, as illustrated in Figure 6, the Compound DAO offers comprehensive documentation, guiding members on how to engage in governance effectively. The absence of such transparency may erode members' trust, thereby discouraging their active participation in DAO governance.

Approach. However, simply adopting basic text searching to check rule satisfaction might introduce false positives, as some documentation may only include keywords as headings without actual content. For example, a DAO from Aragon only mentions, "Governance Proposal This is the last step of the Governance process and is the only one that is binding." In such instances, a basic text search for "Governance process" could result in a false positive. To address this issue, we employ ChatGPT [65] as a question-answering system to determine whether the six rules are truly present in the DAO documentation. Based on recent studies [66][67][68][69], ChatGPT outperforms existing Large Language Models (LLMs) in question-answering tasks. Additionally, it demonstrates superior robustness in question comprehension when compared to state-of-the-art question-answering systems.

Querying a Large Language Model (LLM) with a single complex question can lead to incorrect responses [70]. A

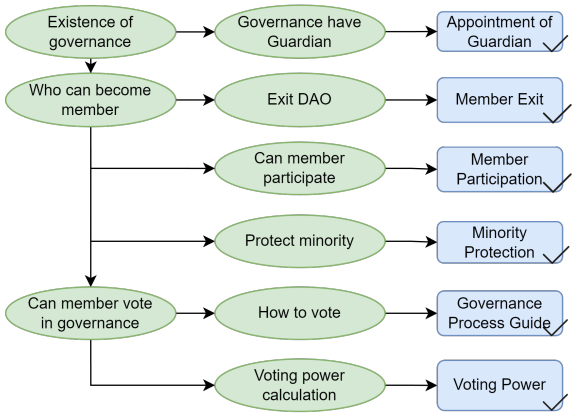


Fig. 7: The abbreviated question chain to query whether the 6 rules are mentioned in the documentation. Each arrow represents a *Yes* response from ChatGPT.

TABLE III: Evaluation of checking whether the rule is mentioned in the documentation.

Rule Name	ChatGPT [65]			Claude [71]		
	Recall	Precision	F1-score	Recall	Precision	F1-score
Member Participation	0.69	0.95	0.80	0.74	0.95	0.83
Member Exit	0.00	0.00	0.00	0.00	0.00	0.00
Voting Power	0.81	0.87	0.84	0.78	0.48	0.59
Minority Protection	1.00	1.00	1.00	1.00	0.50	0.66
Governance Process Guide	0.68	0.90	0.77	0.92	0.80	0.86
Appointment of Guardian	0.89	1.00	0.94	0.89	0.73	0.80

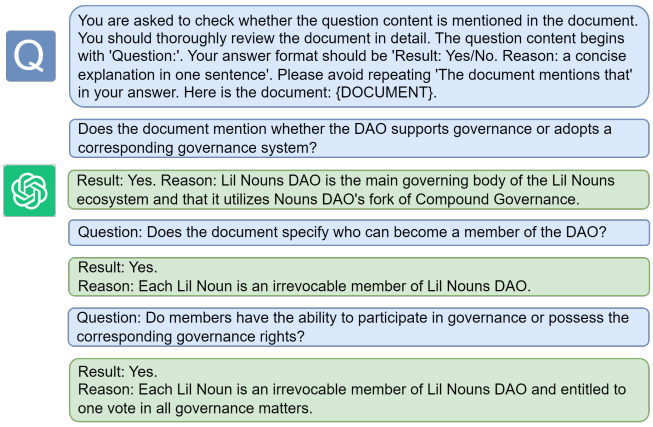


Fig. 8: An example of querying ChatGPT with a chain of questions to check the Rule1 *Member Participation* is mentioned. We remove the cross-verify query for clarity.

TABLE IV: Number of Documentation Satisfying Each Rule

DAO Type	Rule1	Rule2	Rule3	Rule4	Rule5	Rule6
XDAO	10	1	12	0	7	1
Aragon	9	2	9	0	14	4
Tally	21	2	17	0	17	6
DAOhaus	1	0	0	1	1	0
DAOstack	4	0	3	0	2	0
Self-developed	3	0	3	0	3	2
Total	48	5	44	1	44	13

related study [29] suggests that the Chain of Thought (CoT) reasoning method improves LLM comprehension of complex questions. Therefore, we adopt prompt optimization by breaking down the documentation rules into a series of intermediate questions. As shown in Figure 8, to check Rule 1 (i.e., *Member Participation*), we ask three questions: *Does the DAO support governance?*, *Who can become a member of the DAO?*, and *Can members participate in governance?*. If all three questions are confirmed in the documentation, we conclude that the rule is satisfied. For all six rules, we address them with a series of detailed questions derived from the DAO Model Law and merge similar queries to form a question chain, as depicted in Figure 7. We utilize the ChatGPT model *gpt-3.5-turbo-16k-0613* for classification. The prompt for each query is shown in the first box of Figure 8. To cross-verify the results, we use the following prompt to recheck the result: "*Your task is to check if the sentence content is mentioned in the document. Here is the sentence: [REASON]. Your answer format should be: Result: Yes/No. The document is provided below: [DOCUMENT].*" If the two results are inconsistent, we rerun both questions. If the inconsistency persists, we set the final result as "No." If the document exceeds the token limit, we partition it into segments of 12,000 tokens each, with an overlap of 2,000 tokens between segments. The detailed question chain for checking each rule is provided in Appendix C.

Result. To evaluate the effectiveness of ChatGPT, we randomly selected 100 documentation samples and manually analyzed whether each satisfied the specified rules. The results are presented in Table III. The findings demonstrate that ChatGPT, when used with the Chain of Thought (CoT) reasoning method, performs well in determining whether a certain rule is mentioned in the documentation. In addition to ChatGPT, we also evaluated another LLM, Claude [71], for comparison. The results show that Claude can achieve recall rates similar to or even higher than ChatGPT for these queries, suggesting that both LLMs provide sufficiently accurate results. However, Claude produced more false positives than ChatGPT. This higher false positive rate may be due to differences in training data or the possibility that Claude requires a different prompt structure compared to ChatGPT. As a result, we chose ChatGPT to measure the integrity of DAO documentation.

The results of each DAO's documentation and how they align with the rules set by the DAO Model Law are illustrated in Table IV. Our findings reveal that none of the DAO documentation fully complies with all six rules. We found that only five DAOs mentioned Rule 2, *Member Exit*, in their documentation. Further analysis of the DAO Model Law suggests that this rule functions more as a compliance standard rather than a practical guideline for DAOs. In practice, the removal of all tokens belonging to a member is typically considered the default method for member exit from the DAO. As for Rule 4, *Minority Protection*, only one DAO, which belongs to DAOhaus, mentioned it in its documentation. Upon further analysis of the DAOhaus [6] platform, we found that it integrates the *rage quit* procedure into its governance model, ensuring protection for members with less voting power.

To evaluate the concept that well-documented DAOs encourage greater member participation in the governance process, we compare the number of documentation provided by DAOs with their corresponding proposal and voting statistics, as shown in Figure 9 and Figure 10. The results indicate that DAOs with better documentation see significantly higher engagement. Specifically, the average number of proposals drops from 34 to 9, and the average number of voting

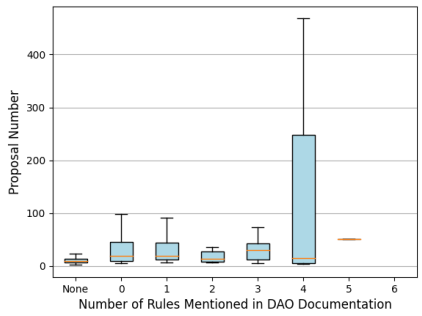


Fig. 9: The number distribution of DAO proposal numbers based on the number of rules satisfied by their documentation

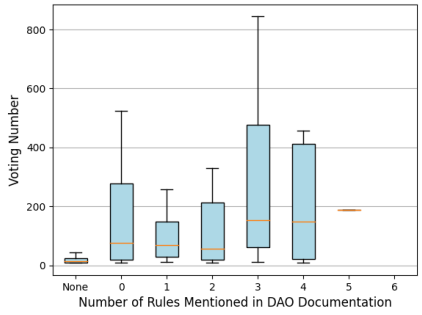


Fig. 10: The number distribution of DAO voting numbers based on the number of rules satisfied by their documentation.

participants declines from 3,342 to just 33 in DAOs without documentation.

Answer to RQ2: Although only 71 out of 3,348 DAOs provide documentation for their members, and none offer complete documentation, more comprehensive documentation significantly helps members actively participate in the DAO governance process.

VII. Does Proposal ensure consistency between descriptions and code?(RQ3)

Proposal has become a primary target for attackers because the proposal's creator can control its actions. This allows attackers to embed malicious code within proposals, aiming to either gain control over the DAO or transfer its assets. In recent years, numerous governance attacks on DAOs have resulted in the loss of millions of dollars [9], [10], [11], [12], [13], [14], [15], [16], [17], [18]. To investigate the security issues in proposals, we first verify the immutability of the proposal code by ensuring that the *target address* in the proposal is open-source and was not created using the $CREATE2$ opcode. Next, we check the consistency between the proposal description and the code by verifying that all actions specified in the code are clearly mentioned in the proposal description.

A. Immutability of Proposal Code

To assess the immutability of the proposal code, we analyze the *target address* within the proposal. The *target address* refers to the contract to be called in the proposal code. It should be open-source so that members can examine the code

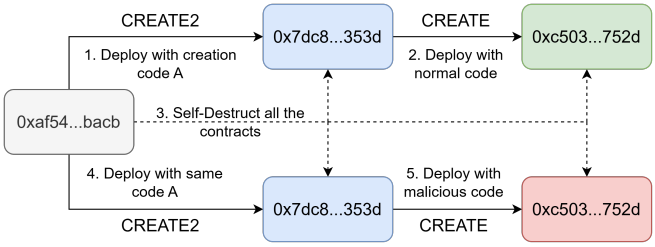


Fig. 11: The example of Tornado cash governance attack. The contract *0xc503...752d* contains proposal code. The attacker uses the *CREATE2* to replace the original code with malicious code before the proposal executed.

logic within the target address. Apart from being open-source, the code logic in the *target address* must also be immutable. As discussed in Section V-C, the EVM opcode $CREATE2$ allows arbitrary change to the code logic inside the *target address* while maintaining the same address. If the proposal code lacks immutability, an attacker can arbitrarily modify the code even after the proposal has been approved. For instance, in the Tornado Cash Governance Attack [13], as illustrated in Figure 11, the attacker first used the *CREATE2* opcode to deploy a contract with the address *0x7dc8...353d*. This contract then deployed another contract, using the *CREATE* opcode, with the address *0xc503...752d*, which contained a normal version of the proposal code. Once the proposal passed but before its execution, the attacker invoked the *SELFDESTRUCT* opcode to destroy both *0x7dc8...353d* and *0xc503...752d*. Subsequently, the attacker redeployed a contract at *0x7dc8...353d* using *CREATE2* with the same creation code, ensuring the redeployed contract retained the same address. Finally, the attacker used this redeployed contract to redeploy the proposal code contract at *0xc503...752d*, this time containing malicious code.

Approach. To assess whether the target address in the proposal code is open-source, we follow the approach used in the previous study [72]. We use APIs provided by blockchain scanners to check if the source code has been verified. We use the same method used in Section V-C to check the *target address* is under the threat of the opcode $CREATE2$. We skip the *target address* that belongs to the governance contract, as it has been evaluated in Section V-C.

Result. The results of the immutability of the proposal code are shown in Table V. We discover that more than 90% (54,108) of the *target address* in the proposal code are open-source. This suggests that the majority of proposals maintain the clarity of their proposal codes. Among the 5,571 closed-source contracts, we identify 32 addresses that have been used by members, as indicated by more than 500 transactions associated with these specific addresses. This implies that some members place their trust in these contracts despite the noticeable lack of transparency. Regarding $CREATE2$, although we do find some target addresses created in the $CREATE2$ chains, they can not destruct themselves and thus are not at risk of being mutated. However, the attacker can insert the $SELFDESTRUCT$ or $DELEGATECALL$ into the target address's code to make this potential threat feasible.

TABLE V: Result of the immutability of target address within proposal code.

DAO Type	Open-source	Close-source	By $CREATE2$	Can $SELFDESTRUCT$
XDAO	28,802	784	8	0
Aragon	17,943	3,080	2	0
Tally	7,384	1,615	97	0
DAOhaus	1,749	78	3	0
DAOstack	2,419	0	0	0
Self-developed	1,382	14	0	0
Total	59,679	5,571	110	0

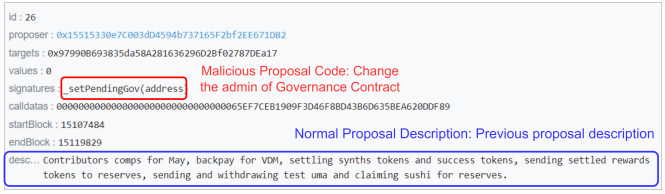


Fig. 12: The malicious proposal in the YAM governance attack deceived members with the description from previous proposal, claiming it would return rewards to the DAO. However, the actual code was to take control of the governance contract.

B. Consistency between Description and Code

The proposal description must fully detail all aspects of the proposal code to ensure that members are well-informed. Otherwise, attackers may hide malicious code within an otherwise normal proposal description. For instance, during the YAM DAO governance attack on July 9, 2022 [9], the attacker submitted a proposal (as shown in Figure 12) with a description copied from a previous proposal, falsely claiming to return rewards to the DAO. In reality, the code transferred ownership of the governance contract to the attacker, resulting in a loss of 3.1 million dollars once the proposal was passed.

Approach. To verify the consistency between the proposal description and code, we first extract the *description intention* from the proposal description and the *code action* from the proposal code. We then check whether the description intention and code action are consistent. The *description intention*—identified as *(action, target object, parameter)*—is derived from the proposal description, outlining the functions intended to be called or not called in the proposal code. The *code action* is extracted from the proposal code, which shows the actual functions to be executed.

1) Description Intention Extractor: The *description intention* is represented as a tuple *(action, target object, parameter)*. The *action* refers to the function name to be performed by the proposal code (e.g., transfer, update, approve), *target object* is the target of the function call, and *parameter* denotes the detailed parameters used by the *action*. We adopt a two-step process to extract the *description intention* from the proposal description. First, We identify all the code-related sentences that describe the function calls in the proposal code. After that, we extract the *description intention* from these code-related sentences based on their grammatical structures. The example procedure of the description intention extractor is shown in Figure 13. The sentence in the red box is identified as code-related. Subsequently, during the intention extraction, the code-related sentence undergoes parsing to form the corresponding semantic dependency parse tree. The *description intention* is then extracted based on the part-of-speech tags and syntactic dependencies in the parse tree.

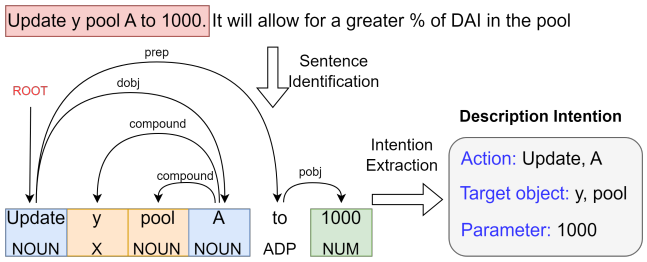


Fig. 13: Example of extracting the *description intention* from the proposal description.

Sentence identification. We apply the NLTK [73] to split the proposal description into individual sentences. In order to identify code-related sentences that describe the code, we use a fine-tuned BERT [74] for the sentence classification task. Due to the absence of a dataset for code-related sentences in DAO proposals, we created a dataset comprising 2,200 sentences randomly extracted from proposals. We select 2,000 sentences from this dataset to fine-tune the BERT model. The remaining 200 sentences are used to evaluate the performance of the find-tuned BERT. We manually annotate each sentence to indicate whether it describes the proposal code. The evaluation of sentence identification is shown in Appendix D.

Intention extraction. To extract the *description intention* from code-related sentences, we first use Spacy [75] to generate a syntactic dependency parse tree and assign part-of-speech (PoS) tags to each token within the sentence. The $action$ is identified by the token that is labeled as $Root$ in the PoS tag. Its lemma either exists in our verb list[1], or it aligns with synonyms of words within our verb list, as determined by the synonyms database [76]. Additionally, the token that has a *direct object (dobj)* relationship with the $Root$ token is also identified in the $action$. The *target object* is identified by tokens that have a $compound$ relationship with the $action$ tokens. Lastly, the $parameter$ is identified by the rest tokens with PoS tags such as $NOUN$, NUM, $PROPN$, or X. As shown in Figure 13, the $action$ is highlighted in the blue box, the *target object* in the yellow box, and the $parameter$ in the green box. We also identify whether the *description intention* originates from negative or positive sentences. To identify these negative sentences, we utilize the BERT to determine whether the code-related sentence is positive or negative. When extracting from these negative sentences, we assign a *Negative* tag to the *description intention*.

2) Code Action Extractor: The code action extractor's purpose is to extract the proposal code and enrich its content, resulting in the *code action* as illustrated in Table VI. Given that the proposal code is in bytecode format, verifying its consistency with the *description intention* could result in false negatives. For instance, in Figure 14, the proposal description outlines its object as *transfer of ARENA tokens*. It is challenging to determine if the code matches the description directly from the bytecode. To address this, we transform the proposal code into *code action* to add natural language information.

Since the proposal code only contains the $target\ address$,

[1]https://drive.google.com/file/d/1I1mPkZMohjC8vINL9JvJSoN8SoymDTRO

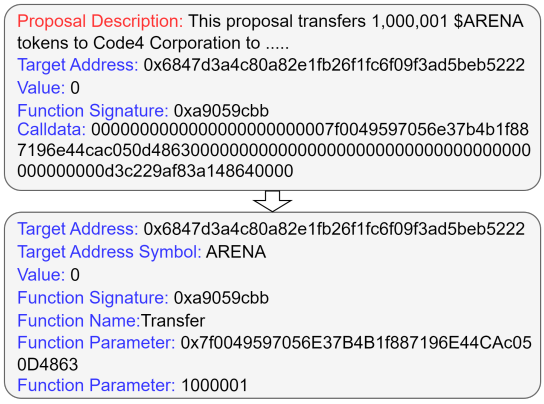


Fig. 14: The example illustrating the extraction and enhancement of proposal code into *code action*.

TABLE VI: Component of *code action.*

Name	Explanation
Target Address	Contract address to be called
Target Address symbol	Contract address name in natural language
Value	Value to transfer
Function Signature	4 bytes of function ID
Function Name	Function name in natural language
Function Parameter	Parameter of the function

$value$, $function\ signature$, and $calldata$ from the proposal code, the rest part of *code action* needs to be enhanced based on these data. The *target address symbol* can be determined by checking the contract address in the public name tag or the function named *symbol()* in the contract. To determine the $function\ name$, we attempt to find it either from the contract ABI [77] of the target address or from the Ethereum Signature Database [78]. We obtain the contract ABI from the source code of the *target address* via blockchain scanner. If the target address is closed-source, we turn to the Ethereum Signature Database [78]—the largest database mapping *function signature* back to *function name*—for querying the $function\ name$. Once the $function\ name$ is extracted, we can decode the $calldata$ into *function parameter*, due to the $function\ name$ containing the sequence and types of each parameter. If we are unable to locate the information, we will leave it empty.

3) Inconsistency Detector: We determine 5 types of inconsistency between the *description intention* and *code action*.

***Lack of* description intention.** We extract the *code action* from the proposal code, but if we are unable to find any corresponding *description intention*, we classify the proposal as lacking *description intention*. Specifically, if we fail to extract the description intention from the proposal description or if the description is missing altogether, we determine that the proposal lacks *description intention*.

***Lack of* code action.** We extract the *description intention* from the proposal description but fail to find any corresponding *code action*. We check whether the proposal contains the proposal code. If it does not, we determine the lack of *code action*.

Incomplete function. In this type of inconsistency, the function present in the *code action* is not mentioned in the *description intention*. To address this, we compare the semantic similarity between the *code action* and the *description intention*. We first extract the *Target Address Symbol* and *Function Name* from the *code action*, and the *Target Object* and *Action* from the *description intention*. Then, we use the Sentence-BERT model [79] to encode the corresponding sets into semantic vectors. We calculate the cosine similarity between these vectors. If the semantic similarity score exceeds the threshold, we conclude that the function is mentioned in the description, as the description conveys a similar semantic meaning. The threshold value is set at 0.75, following the official examples from Sentence-BERT [80].

Incomplete parameter. For this type of inconsistency, we verify that every *Function Parameter* listed in the *code action* is described in the *Parameter* section of the *description intention*. We assess parameters of the following types: address, number, and byte, as well as their corresponding lists. For the address type, we first retrieve its name using the method described in the information enhancement section. If we can extract the name, we check whether it appears in the *Parameter* section of the *description intention*. If the name is not found, we directly verify whether the address, in hexadecimal format, is mentioned. For the number type, we check if the number is present in the *description intention*. If the target contract is an ERC-20, we adjust the value by dividing it by its decimals, following the ERC-20 standard [81]. For the byte type, since it can represent text in hexadecimal form, we check whether both the original content and its decoded text appear in the *description intention*. we determine that the parameter is incomplete if is not mentioned in the *description intention*.

Incorrect proposal. The incorrect proposal is determined by identifying whether a function in the *code action* is mentioned by a *description intention* tagged as *Negative*. We use the same method applied in detecting incomplete functions by checking whether the semantic similarity between the function in the *code action* and the *description intention* (tagged as *Negative* by *Intention extraction*) exceeds the threshold.

TABLE VII: Evaluation results for Inconsistency Detector.

Incomplete Type	Precision	Recall	F-1
Incomplete Code Action Function	0.81	0.87	0.84
Incomplete Code Action Parameter	0.83	0.88	0.85

To evaluate the performance of our *Inconsistency Detector*, we randomly selected 1,500 functions, containing 3,122 parameters. We manually labeled the results for each function and parameter. The evaluation results are shown in Table VII. For incomplete functions, the detector achieved a precision of 0.81 and a recall of 0.87. False positives occurred because BERT struggled to correctly interpret the relationships between words with similar semantic meanings, particularly when descriptions contained only a few nouns. False negatives arose due to differences in word meanings between the real world and the blockchain context. For incomplete parameters, we achieved a precision of 0.83 and a recall of 0.88. False positives were caused when proposal descriptions used URLs to describe function parameters, while false negatives occurred due to misleading parameter names.

Result. The results of the 5 types of inconsistency are shown in Table VIII. We excluded proposals from platform XDAO and the self-developed MakerDAO, as these 2 types of DAOs do not support proposals that include descriptions. Instead, they require members to submit code directly to the

TABLE VIII: Result of consistency between description and code. The *description intention* is short for DI and *code action* is short for CA.

Consistency Type	Aragon	Tally	DAOhaus	DAOstack	Self-developed	Total
Normal	968	1,328	12	671	39	3,018
Lack of DI	18,581	2,401	698	29	333	22,042
Lack of CA	893	38	0	1,453	0	2,384
Incomplete	581	5,232	1,117	266	1,072	8,268
– Function	659	7,010	1,130	212	504	9,515
– Parameter	17,166	23,451	3,065	375	1,450	45,507
Incorrect	0	0	0	0	0	0
Proposal	21,023	8,999	1,827	2,419	1,250	35,518
Function	20,273	25,184	3,175	616	2,614	51,862
Parameter	30,424	33,951	3,513	745	2,803	71,436

TABLE IX: Classification of the collected real-world governance attack incidents. The *description intention* is short for DI and *code action* is short for CA.

Incidents	Date	Attack Result	Expect Lost	Proposal Consistency
True Seigniorage Dollar [14]	Mar 2021	Successed	$16K	Lack of DI
Yuan [17]	Sep 2021	Successed	$250K	Lack of DI
Venus [86]	Sep 2021	Successed	$250K	Lack of DI
Build Finance [16]	Feb 2022	Successed	$470K	Lack of DI
Fortress Protocol [12]	May 2022	Successed	$3M	Incomplete Parameter
Beanstalk [18]	Apr 2022	Successed	$182M	Incomplete Function
Audius [11]	Jul 2022	Successed	$1.1M	Lack of DI
YAM [9]	Jul 2022	Blocked	$2.1M	Incomplete Function
Swerve Finance [87]	Mar 2023	Successed	$1.3M	Lack of DI
Tornado Cash [13]	May 2023	Successed	$2M	Code Mutability
Atlantis Loans [88]	Jun 2023	Successed	$1M	Lack of DI
BIGCAP [90]	Sep 2023	Blocked	$45K	Incomplete Function
Indexed Finance [89]	Nov 2023	Blocked	$158K	Lack of DI

governance contract as a proposal, which could interfere with the accuracy of our analysis. Our results suggest that members currently do not pay sufficient attention to proposals. Of the 35,518 proposals analyzed, 24,426 either lack a description of the proposal code or only contain a description without corresponding code. Furthermore, among the 11,092 proposals that do include both a description and code, 8,268 are found to be incomplete, either lacking an explanation about the functions or detailed parameters in the functions.

Real-World Attack Cases Detection. To assess whether our approach is capable of detecting real-world malicious proposals, we have gathered reports of DAO governance attack cases from the following sources: Slowmist [82], CryptoSec [83], Rekt [84], and Twitter [85]. We total collected 11 DAO governance attack cases [17], [14], [16], [18], [11], [86], [9], [12], [87], [13], [88], [89], [90]. Upon examining these malicious proposals with our approach, we identified all 13 proposals as 8 malicious proposals due to lack of *description intention*, 3 proposals due to incomplete function, 2 proposals with incomplete parameter, and 1 proposal is subjected to mutability of proposal code.

Answer to RQ3: Although most proposal code is open-source and can be reviewed by members, approximately 10% (5,571) of proposal code is closed-source, making it difficult for members to scrutinize. Among the 35,518 proposals analyzed, 32,500 (about 91%) fail to provide consistent descriptions and corresponding code. This inconsistency highlights why attackers frequently target proposals during the governance process.

VIII. Discussion

A. Threat to Validity

Complete DAO data. We have employed the following method to collect a comprehensive DAO dataset. First, we collect DAO data from various sources, including previous studies [36], [26], as well as from well-known industry dataset DeepDAO [8]. Second, we expand our collection to include DAOs from Ethereum and 8 other popular blockchains. Third, we gather data from both websites and blockchains to ensure the data completeness. As a result, our dataset, comprising over 3,000 DAOs, 200 websites, and 65,000 proposals, is the most comprehensive DAO dataset to date. The findings derived from this dataset can be considered representative of the entire DAO ecosystem. However, there may still be some self-developed DAOs or platforms that were not captured. Our approach can be applied to such DAOs once they provide their governance contract address and documentation website.

DAOs from non-EVM-compatible chains. According to statistics from DefiLlama [91], EVM-compatible chains currently dominate the blockchain ecosystem, accounting for over 85% of the Total Value Locked (TVL) across all blockchains. Therefore, we primarily applied our approach to EVM-compatible chains. However, aside from the immutability of contracts, our approach and insights are not solely dependent on EVM-specific features. Thus, our approach can be applied to non-EVM-compatible chains as well.

Off-chain governance DAOs. In off-chain governance, the governance process takes place on the website, where members submit proposals and cast their votes. The execution of these proposals is carried out by the DAO developers rather than being automatically triggered by smart contracts [32], [33]. According to the definition of DAOs provided by Ethereum [1] and the DAO Model Law [30], DAOs must be governed by smart contracts. Therefore, off-chain governance DAOs fall outside of our scope.

B. Limitations

Querying DAO Documentation In Section VI (RQ2), our method leverages LLM to verify whether the provided DAO documentation aligns with the requirements outlined in the DAO model laws. However, due to current token limitations in LLMs, large documents must be divided into smaller segments, and full-length rule descriptions with detailed explanations from the DAO model laws cannot be directly utilized. To address these challenges, we integrate CoT reasoning to enhance the performance of LLMs. Despite these efforts, advancements in LLMs that support larger content sizes, combined with the application of prompt engineering techniques, are anticipated to improve the perform of semantic search.

Proposal Description and Code Consistency In Section VII-B (RQ3), our method evaluates the consistency between a proposal's description and its code by extracting the description intent and the code actions, then identifying 5 types of inconsistencies. However, this approach may result in some loss of information from both the code and the description. To address this limitation, we could involve fine-tuning LLM

using the current inconsistency results to improving the ability to detect inconsistencies with greater accuracy.

IX. Implications and suggestions.

Based on our research findings, we recommend that DAO platforms ensure all DAOs established on their platforms adhere to the principles of decentralized governance, rather than allowing developers to optionally support it. Developers should be required to disclose all privileged addresses to their members or mandate that all privileged functions be controlled by the governance contract. Additionally, they should provide complete documentation to facilitate member participation in the governance process. Blockchain scanners, such as Etherscan, should label contracts that are deployed using the *CREATE2* opcode. In response to the observed inconsistencies in proposals, we suggest that DAOs enforce consistency between proposal descriptions and the actual code. Additionally, tools should be developed to automatically supplement proposal descriptions with any missing code and explanations.

X. Feature works

Automate DAO Reinforcement Our method efficiently and accurately identifies issues within the governance process. However, it currently lacks the capability to automatically generate patches to address these issues. Future work could integrate static analysis techniques and LLM to automate patch generation for governance contracts. This approach could also be extended to automatically generate the required DAO documentation, ensuring sufficient and accurate documentation for all six rules. Additionally, the method could automate the completion of proposal descriptions based on the provided proposal code, fostering a more robust and transparent DAO governance ecosystem.

Governance Procee Attack Detection Our work identifies several vulnerabilities within the DAO governance process, such as privileged functions in governance contracts and inconsistencies between proposal descriptions and their code. Future work could leverage these identified issues to develop tools that help DAO developers and members detect malicious DAOs or proposals. Such tools could play a critical role in preventing prevalent attacks against DAOs, enhancing security and trust in decentralized governance frameworks.

XI. Related Work

DAO. Recent research on DAO focuses on the DAO activity analysis [20], [21], [7], [92], [93], [22], DAO definition and application [94], [95], and DAO governance method [28], [96], [97]. However, they do not concentrate on the security aspects of DAO governance. As for empirical studies that do focus on security within DAO governance: Feichtinger *et al.*[24] provided analysis on 21 on-chain governance DAOs, specifically focusing on the voting process within the governance procedure. Fritsch *et al.*[27] focused on the distribution of voting power among three popular DAOs: Compound, Uniswap, and ENS. Sharma *et al.*[25] analyzed the existing centralized risk of 10 existing DAOs and the corresponding members voting behaviors. Wang *et al.*[26] analyzed the design principles of DAOs from off-chain voting platform Snapshot. Liu *et al.*[23] focused on voting behavior in DAO governance. Dotan *et al.*[22] disclosed the centralized voting nature of four DAOs and explained the existing governance attack incidents. The above research primarily focused on partial aspects such as voting within the DAO governance framework, and their datasets are limited, no larger than 1,000 DAOs. Our methodology analyzes the security issues across both on-chain and off-chain parts of the governance framework. The security threats we studied have not been explored in previous research.

Smart contracts analysis. Smart contracts have gained popularity for facilitating trustless code execution on the blockchain. However, with the increasing usage of smart contracts, they have become targets for attacks. Numerous tools have been developed for the analysis of smart contracts. Some notable examples include Mythril [98], Manticore [99], and Oyente [100]. Pied-Piper [58] proposed a hybrid analysis method that combines datalog analysis and directed fuzzing to detect potential backdoor threats in ERC token contracts in order to enhance smart contract security. Beyond the direct analysis of bytecode, binary lifter tools such as Gigahorse [101] transform the bytecode into a higher-level, function-based, three-address representation. Our method targets the detection of security issues within governance contracts and can be integrated with existing tools to enhance the security of dApps.

Consistency between code and natural language description. The consistency between the code and natural language description has been well-studied [102], [103], [104], [105], [106], [107]. They primarily concentrate on Java code and API documentation, which are well-written and focused on describing code behavior. DocCon [108] detects inconsistencies between documentation and the corresponding code for Solidity smart contract libraries. Compared with Doccon, our method targets different research questions. Our natural language description comes from proposal description, which lacks structured information such as tags in the comments or API document. Additionally, the proposal description encompasses a broader scope instead of only describing the code behavior. The code in our method is the bytecode, not the Solidity source code, which lacks code information like variable name. Furthermore, our code size is extremely limited, containing only several bytes and the function call parameters rather than the full code logic.

XII. Conclusion

In this paper, we conduct a comprehensive study of the issues in the DAO governance process components. We construct the dataset contains 3,348 DAOs, 144 documentation, and 65,436 proposals across 9 different blockchains. Then we apply our novel methods to automatically identifying issues within these components. For Impartial Decentralized Governance in the Governance Contract, we found that out of the 3,348 DAOs analyzed, 176 could potentially be manipulated by developers, with one DAO's governance contract logic being directly alterable by its developer. This suggests that not all active DAOs can be trusted to maintain impartial

decentralized governance. For Sufficient Governance Process Documentation, only 71 out of 3,348 DAOs provide any form of documentation for their members, and none offer complete documentation. However, more comprehensive documentation significantly enhances member participation in the DAO governance process. Finally, for Proposal Consistency, while most proposal code is open-source and available for review by members, approximately 10% (5,571) of target address within proposal code are closed-source, making them difficult for members to scrutinize. Among the 35,518 proposals analyzed, 32,500 (about 91%) fail to provide consistent descriptions and corresponding code. This might explain why attackers frequently target proposals during the governance process.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their insightful comments and valuable feedback. This work is supported by the National Natural Science Foundation of China (No. 62372218 , No. U24A6009, No. 62172301) and Hong Kong RGC Projects (PolyU15224121, PolyU15231223).

REFERENCES

[1] V. Buterin *et al.*, "A next-generation smart contract and decentralized application platform," *white paper*, 2014.
[2] "Uniswap." https://uniswap.org/, 2023.
[3] "CoinMarketCap." https://coinmarketcap.com/view/dao/, 2023.
[4] "XDAO." https://docs.xdao.app/, 2023.
[5] "Aragon." https://legacy-docs.aragon.org/aragon/readme, 2023.
[6] "Daohaus." https://daohaus.club/, 2023.
[7] Y. Faqir-Rhazoui, J. Arroyo, and S. Hassan, "A comparative analysis of the platforms for decentralized autonomous organizations in the ethereum blockchain," *Journal of Internet Services and Applications*, 2021.
[8] "Deepdao." https://deepdao.io/organizations, 2023.
[9] "Yam attack analysis." https://decrypt.co/104848/yam-finance-safeguards-3-1m-treasury-governance-attack, 2023.
[10] "Potential curve dao attack." https://gov.curve.fi/t/the-curve-emergency-dao-has-killed-the-usdm-gauge/2307, 2023.
[11] "Audius dao attack." https://cointelegraph.com/news/hackerdrains-1-08m-from-audius-following-passing-of-malicious-proposal, 2023.
[12] "Fortress protocol attack." https://rekt.news/fortress-rekt/, 2023.
[13] "Attacker hijacks Tornado Cash governance via malicious proposal." https://cointelegraph.com/news/attacker-hijacks-tornado-cash-governance-via-malicious-proposal, 2024.
[14] "True seigniorage dollar attack." https://twitter.com/TrueSeigniorage/status/1370956726489415683, 2023.
[15] "Pride punks dao attack." https://twitter.com/BoringSecDAO/status/1556150989140373504, 2023.
[16] "Build Finance suffers from governance attack." https://cryptoslate.com/build-finance-dao-hostile-takeover-treasury-drained/, 2023.
[17] "Yuan.finance attack report." https://medium.com/yuan-finance/yuan-governance-attack-update-and-migration-plan-3b5d949ab466, 2023.
[18] "Beanstalk Exploit — A Simplified Post-Mortem Analysis." https://medium.com/coinmonks/beanstalk-exploit-a-simplified-post-mortem-analysis-92e6cdb17ace, 2023.
[19] "VPANDA DAO Rug Pull." https://twitter.com/DeDotFiSecurity/status/1669859985113731082, 2023.
[20] Y. Faqir-Rhazoui, M.-J. Ariza-Garzón, J. Arroyo, and S. Hassan, "Effect of the gas price surges on user activity in the daos of the ethereum blockchain," in *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, 2021.
[21] Y. El Faqir, J. Arroyo, and S. Hassan, "An overview of decentralized autonomous organizations on the blockchain," in *Proceedings of the 16th international symposium on open collaboration*, pp. 1–8, 2020.
[22] M. Dotan, A. Yaish, H.-C. Yin, E. Tsytkin, and A. Zohar, "The vulnerable nature of decentralized governance in defi," in *Proceedings of the 2023 Workshop on Decentralized Finance and Security*, 2023.
[23] X. Liu, "The illusion of democracy? an empirical study of dao governance and voting behavior," *An Empirical Study of DAO Governance and Voting Behavior (May 8, 2023)*, 2023.
[24] R. Feichtinger, R. Fritsch, Y. Vonlanthen, and R. Wattenhofer, "The hidden shortcomings of (d) aos–an empirical study of on-chain governance," *arXiv preprint arXiv:2302.12125*, 2023.
[25] T. Sharma, Y. Kwon, K. Pongmala, H. Wang, A. Miller, D. Song, and Y. Wang, "Unpacking how decentralized autonomous organizations (daos) work in practice," *arXiv preprint arXiv:2304.09822*, 2023.
[26] Q. Wang, G. Yu, Y. Sai, C. Sun, L. D. Nguyen, S. Xu, and S. Chen, "An empirical study on snapshot daos," *arXiv preprint arXiv:2211.15993*, 2022.
[27] R. Fritsch, M. Müller, and R. Wattenhofer, "Analyzing voting power in decentralized governance: Who controls daos?," *arXiv preprint arXiv:2204.01176*, 2022.
[28] T. Dursun and B. B. Üstündağ, "A novel framework for policy based on-chain governance of blockchain networks," *Information Processing & Management*, 2021.
[29] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in Neural Information Processing Systems*, 2022.
[30] "DAO Model Law." https://coala.global/daomodellaw/, 2023.
[31] "Compound documents." https://docs.compound.finance/v2/, 2023.
[32] W. Reijers, I. Wuisman, M. Mannan, P. De Filippi, C. Wray, V. Rae-Looi, A. Cubillos Vélez, and L. Orgad, "Now the code runs itself: On-chain and off-chain governance of blockchain technologies," *Topoi*, 2021.
[33] P. De Filippi and G. McMullen, *Governance of blockchain systems: Governance of and by Distributed Infrastructure*. PhD thesis, Blockchain Research Institute and COALA, 2018.
[34] "The Model Law." https://uncitral.un.org/en/texts/arbitration/modellaw/commercial_arbitration, 2023.
[35] "Synthetify governance attack." https://blockworks.co/news/solana-exploit-dao-hacker, 2024.
[36] J. Arroyo, D. Davó, E. Martínez-Vicente, Y. Faqir-Rhazoui, and S. Hassan, "Dao-analyzer: Exploring activity and participation in blockchain organizations," in *Companion Publication of the 2022 Conference on Computer Supported Cooperative Work and Social Computing*, pp. 193–196, 2022.
[37] "Daostack." https://daostack.io/, 2023.
[38] "Tally." https://www.tally.xyz/, 2023.
[39] "Ethereum." https://ethereum.org/en/, 2024.
[40] "BSC." https://www.bnbchain.org/en, 2024.
[41] "Polygon." https://www.polygon.com/, 2024.
[42] "Fantom." https://fantom.foundation/, 2024.
[43] "Gnosis." https://www.gnosis.io/, 2024.
[44] "Avalanche." https://www.avax.network/, 2024.
[45] "Arbitrum." https://arbitrum.io/, 2024.
[46] "Cronos." https://cronos.org/, 2024.
[47] "Optimism." https://www.optimism.io/, 2024.
[48] "Curve." https://curve.fi/, 2023.
[49] "MakerDAO." https://makerdao.com/en/, 2023.
[50] "Public name tags." https://info.etherscan.com/public-name-tags-labels/, 2023.
[51] "Selenium." https://www.selenium.dev/, 2023.
[52] "Event logs." https://info.etherscan.com/what-is-event-logs/, 2023.
[53] "How to set up on-chain governance." https://docs.openzeppelin.com/contracts/4.x/governance, 2023.
[54] "Evm cfg builder." https://github.com/crytic/evm_cfg_builder, 2023.
[55] J. Xu, K. Paruch, S. Cousaert, and Y. Feng, "Sok: Decentralized exchanges (dex) with automated market maker (amm) protocols," *ACM Computing Surveys*, vol. 55, no. 11, pp. 1–50, 2023.
[56] L. Zhou, X. Xiong, J. Ernstberger, S. Chaliasos, Z. Wang, Y. Wang, K. Qin, R. Wattenhofer, D. Song, and A. Gervais, "Sok: Decentralized finance (defi) attacks," in *2023 IEEE Symposium on Security and Privacy (SP)*, pp. 2444–2461, IEEE, 2023.
[57] M. Fröwis and R. Böhme, "Detecting privileged parties on ethereum," 2022.
[58] F. Ma, M. Ren, L. Ouyang, Y. Chen, J. Zhu, T. Chen, Y. Zheng, X. Dai, Y. Jiang, and J. Sun, "Pied-piper: Revealing the backdoor threats in ethereum erc token contracts," *ACM Transactions on Software Engineering and Methodology*, 2023.

[59] "Ethereum Constantinople/St. Petersburg Upgrade Announcement." https://blog.ethereum.org/2019/02/22/ethereum-constantinople-st-petersburg-upgrade-announcement, 2023.

[60] "Eip-1014: Skinny create2." https://eips.ethereum.org/EIPS/eip-1014, 2023.

[61] M. Fröwis and R. Böhme, "Not all code are create2 equal," in *6th Workshop on Trusted Smart Contracts (WTSC'22)*, 2022.

[62] "Proxy contracts." https://info.etherscan.com/what-is-proxy-contract/, 2023.

[63] "Tenderly." https://tenderly.co/, 2023.

[64] S. Boss, "Daos: Legal and empirical review," *Blockchain & Society Policy Research Lab Research Nodes*, 2023.

[65] "Chatgpt." https://openai.com/blog/chatgpt, 2023.

[66] Y. Tan, D. Min, Y. Li, W. Li, N. Hu, Y. Chen, and G. Qi, "Evaluation of chatgpt as a question answering system for answering complex questions," *arXiv preprint arXiv:2303.07992*, 2023.

[67] N. Bian, X. Han, L. Sun, H. Lin, Y. Lu, and B. He, "Chatgpt is a knowledgeable but inexperienced solver: An investigation of commonsense problem in large language models," *arXiv preprint arXiv:2303.16421*, 2023.

[68] Q. Zhong, L. Ding, J. Liu, B. Du, and D. Tao, "Can chatgpt understand too? a comparative study on chatgpt and fine-tuned bert," *arXiv preprint arXiv:2302.10198*, 2023.

[69] R. Omar, O. Mangukiya, P. Kalnis, and E. Mansour, "Chatgpt versus traditional question answering for knowledge graphs: Current status and future directions towards knowledge graph chatbots," *arXiv preprint arXiv:2302.06466*, 2023.

[70] S. Zheng, J. Huang, and K. C.-C. Chang, "Why does chatgpt fall short in answering questions faithfully?," *arXiv preprint arXiv:2304.10513*, 2023.

[71] "Claude." https://claude.ai/, 2023.

[72] D. Das, P. Bose, N. Ruaro, C. Kruegel, and G. Vigna, "Understanding security issues in the nft ecosystem," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022.

[73] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[74] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[75] "Spacy." https://github.com/explosion/spaCy, 2023.

[76] "Synonym." https://www.synonym.com/, 2023.

[77] "Contract ABI Specification." https://docs.soliditylang.org/en/v0.8.19/abi-spec.html, 2023.

[78] "Ethereum Signature Database." https://www.4byte.directory/, 2023.

[79] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *arXiv preprint arXiv:1908.10084*, 2019.

[80] "Official sentence-bert examples." https://github.com/UKPLab/sentence-transformers/blob/master/examples/app-lications/clustering/fast_clustering.py#L57, 2023.

[81] "ERC-20 Token Standard." https://eips.ethereum.org/EIPS/eip-20, 2023.

[82] "SlowMist." https://www.slowmist.com/, 2023.

[83] "CryptoSec." https://cryptosec.info/, 2023.

[84] "rekt." https://rekt.news/, 2023.

[85] "Twitter." https://twitter.com/home, 2023.

[86] "Venus protocol prevented hostile takeover attempt." https://www.cryptotimes.io/venus-protocol-prevented-hostile-takeover-attempt/, 2023.

[87] "Defunct swerve finance still subject of 1.3 million live governance hack." https://www.theblock.co/post/222744/defunct-swerve-finance-still-subject-of-1-3-million-live-governance-hack, 2023.

[88] "Atlantis loans hack analysis." https://blog.solidityscan.com/atlantis-loans-hack-analysis-7f3fb2e295e0, 2023.

[89] "Indexed finance dao attack." https://blockworks.co/news/blackmail-thwarts-90k-dao-attack, 2024.

[90] "Bigcap dao attack." https://twitter.com/BIGCAPProject/status/1697958233204490494, 2024.

[91] "Total value locked all chains." https://defillama.com/chains, 2023.

[92] O. Rikken, M. Janssen, and Z. Kwee, "The ins and outs of decentralized autonomous organizations (daos)," *Available at SSRN 3989559*, 2018.

[93] X. Zhao, P. Ai, F. Lai, X. Luo, and J. Benitez, "Task management in decentralized autonomous organization," *Journal of Operations Management*, 2022.

[94] E. Baninemeh, S. Farshidi, and S. Jansen, "A decision model for decentralized autonomous organization platform selection: Three industry case studies," *arXiv preprint arXiv:2107.14093*, 2021.

[95] L. Liu, S. Zhou, H. Huang, and Z. Zheng, "From technology to society: An overview of blockchain-based dao," *IEEE Open Journal of the Computer Society*, 2021.

[96] C. Calcaterra, "On-chain governance of decentralized autonomous organizations: Blockchain organization using semada," *Available at SSRN 3188374*, 2018.

[97] X. Fan, Q. Chai, and Z. Zhong, "Multav: A multi-chain token backed voting framework for decentralized blockchain governance," in *International Conference on Blockchain*, 2020.

[98] B. Mueller, "Smashing ethereum smart contracts for fun and real profit," *HITB SECCONF Amsterdam*, 2018.

[99] M. Mossberg, F. Manzano, E. Hennenfent, A. Groce, G. Grieco, J. Feist, T. Brunson, and A. Dinaburg, "Manticore: A user-friendly symbolic execution framework for binaries and smart contracts," in *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 1186–1189, IEEE, 2019.

[100] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor, "Making smart contracts smarter," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016.

[101] N. Grech, L. Brent, B. Scholz, and Y. Smaragdakis, "Gigahorse: thorough, declarative decompilation of smart contracts," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pp. 1176–1186, IEEE, 2019.

[102] L. Yu, X. Luo, J. Chen, H. Zhou, T. Zhang, H. Chang, and H. K. Leung, "Ppchecker: Towards accessing the trustworthiness of android apps' privacy policies," *IEEE Transactions on Software Engineering*, 2018.

[103] B. Andow, S. Y. Mahmud, W. Wang, J. Whitaker, W. Enck, B. Reaves, K. Singh, and T. Xie, "{PolicyLint}: investigating internal privacy policy contradictions on google play," in *28th USENIX security symposium (USENIX security 19)*, 2019.

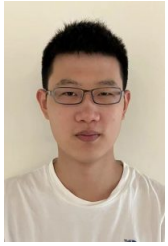
[104] D. Torre, S. Abualhaija, M. Sabetzadeh, L. Briand, K. Baetens, P. Goes, and S. Forastier, "An ai-assisted approach for checking the completeness of privacy policies against gdpr," in *2020 IEEE 28th International Requirements Engineering Conference (RE)*, 2020.

[105] H. Zhong and Z. Su, "Detecting api documentation errors," in *Proceedings of the 2013 ACM SIGPLAN international conference on Object oriented programming systems languages & applications*, 2013.

[106] Y. Zhou, R. Gu, T. Chen, Z. Huang, S. Panichella, and H. Gall, "Analyzing apis documentation and code to detect directive defects," in *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*, 2017.

[107] Y. Zhou, C. Wang, X. Yan, T. Chen, S. Panichella, and H. Gall, "Automatic detection and repair recommendation of directive defects in java api documentation," *IEEE Transactions on Software Engineering*, 2018.

[108] C. Zhu, Y. Liu, X. Wu, and Y. Li, "Identifying solidity smart contract api documentation errors," in *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, 2022.



Junjie Ma received the Bachelor degree in Computer Science and Technology from Northeastern University, China in 2022. He is currently working towards the Ph.D. degree under the collaborative Ph.D. program in the Department of Computer Science and Engineering at Southern University of Science and Technology and the Department of Computing at The Hong Kong Polytechnic University, under the supervision of Prof. Daniel Xiapu Luo, Prof. Wang Qi, and Prof. Fengwei Zhang. His current research interests include decentralized autonomous organizations and blockchain security.

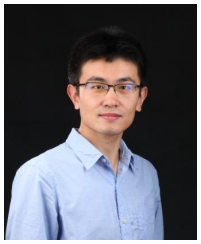
Muhui Jiang obtained his Ph.D. degree in Department of Computing from the the Hong Kong Polytechnic University. Before coming to PolyU, He received his B.Eng. in Department of Software Engineering, Tongji University in 2016. His current research interests include blockchain security, network security, system security and IoT security. More specifically, He is interested in reverse engineering, binary analysis, firmware rehosting, fuzzing techniques, and Defi security.

Jinan Jiang obtained his B.A. degree in Computer Science from University of California, Berkeley. He is currently working towards the Ph.D. degree in the Department of Computing, The Hong Kong Polytechnic University, under the supervision of Prof. Daniel Xiapu Luo. His current research interests include smart contract analysis and blockchain security.

Xiapu Luo is a professor at the Department of Computing of the Hong Kong Polytechnic University. His research focuses on Blockchain and Smart Contracts Security, Mobile and IoT Security, Network Security and Privacy, and Software Engineering with papers published in top-tier security, software engineering, and networking venues. His research led to more than ten best/distinguished paper awards, including ACM CCS'24 Distinguished Paper Award, three ACM SIGSOFT Distinguished Paper Awards in ICSE'24, ISSTA'22 and ICSE'21, Best DeFi Papers Award 2023, Best Paper Award in INFOCOM'18, Best Research Paper Award in ISSRE'16, etc. and several awards from the industry. He received the BOCHK Science and Technology Innovation Prize (FinTech) for his contribution to blockchain security. He regularly serves in the program committees of top security and software engineering conferences and received Top Reviewer Award from CCS'22 and Distinguished TPC member Award from INFOCOM'23 and INFOCOM'24.

Yufeng Hu received the Bachelor degree in mathematics and finance from Zhejiang University, in 2020. He is currently working towards a PhD degree in cyberspace Security at Zhejiang University. His research interests include blockchain security, smart contract security, anti-money laundering, and binary security.

Yajin Zhou is a ZJU 100-Young professor (since 2018), with both the College of Computer Science and Technology and the School of Cyber Science and Technology at Zhejiang University, China. He earned his Ph.D. (2015) in Computer Science from North Carolina State University (Advisor: Prof. Xuxian Jiang), and then worked as a senior security researcher at Qihoo 360. He has published more than 40 papers, with 7500+ citations (Google Scholar). Two of his papers have been selected to the list of normalized Top-100 security papers since 1981. He was recognized as the Most Influential Scholar Award Honorable Mention for his contributions to the field of Security and Privacy (Rank 48 from 2010 - 2019, Rank 6 from 2011 - 2020). His joint team with City University of Hong Kong won first place in the 2019 iDash competition (SGX Track). His current research spans software security, operating systems security, hardware-assisted security and confidential computing. He is also interested in emerging areas, e.g., security of smart contracts, decentralized finance (DeFi) security, and underground economy.



Qi Wang received the B.Eng. degree from the University of Science and Technology of China (USTC) in 2007 and the Ph.D. degree from The Hong Kong University of Science and Technology (HKUST) in 2011. From October 2011 to September 2013, he was an Alexander von Humboldt Post-Doctoral Researcher with Otto-von-Guericke University Magdeburg, Magdeburg, Germany. From October 2013 to September 2014, he was a Research Associate with HKUST. He has been with the Department of Computer Science and Engineering, Southern University of Science and Technology, since 2014, where he is currently a tenured Associate Professor. His research interests include coding theory, cryptography, and combinatorial designs.

Fengwei Zhang received the Ph.D. degree in computer science from George Mason University. He is currently an Associate Professor with the Department of Computer Science and Engineering, Southern University of Science and Technology (SUSTech). His research interests include systems security, with a focus on trustworthy execution, hardware-assisted security, debugging transparency, transportation security, and plausible deniability encryption.